**Аннотация**

Целью дипломной работы является разработка механизма прямых трансграничных финансовых операций между цифровыми валютами разных стран.

Дипломный проект выполнен в Microsoft Word 2021, код программы написан средствами текстового редактора PyCharm.

Работа содержит введение, n глав, заключение, библиографический список и приложение.

Данная выпускная квалификационная работа содержит: листов – n, рисунков - n, таблиц - n, формул - n, приложений - n, источников использованной литературы - n.

**Содержание**

[**Определения, обозначения и сокращения** 4](#_Toc165476410)

[**Введение** 7](#_Toc165476411)

[**Глава 1. Анализ предметной области** 11](#_Toc165476412)

[**1.1** **Изучение существующих аналогов блокчейнов** 11](#_Toc165476413)

[**1.2 Анализ потребностей и проблем, которые приложение должно решать** 14](#_Toc165476414)

[**1.3 Выбор среды разработки** 16](#_Toc165476415)

[**1.4 Изучение возможностей фреймворка FastAPI** 19](#_Toc165476416)

[**1.4.1 Преимущества фреймворка FastAPI** 20](#_Toc165476417)

[**1.4.2 Недостатки фреймворка FastAPI** 21](#_Toc165476418)

[**1.5 Выводы по главе 1** 22](#_Toc165476419)

[**Глава 2. Проектирование приложения** 23](#_Toc165476420)

[**2.1 Определение требований к приложению** 23](#_Toc165476421)

[**2.2 Разработка структуры приложения и его компонентов** 25](#_Toc165476422)

[**2.3 Разработка интерфейса пользователя** 29](#_Toc165476423)

[**2.4 Описание архитектуры приложения** 31](#_Toc165476424)

[**2.5 Выводы по главе 2** 34](#_Toc165476425)

[**Глава 3. Разработка приложения** 35](#_Toc165476426)

[**3.1 Создание хранилища для блокчейна и его описание** 35](#_Toc165476427)

[**3.2 Реализации логики работы блокчейна** 40](#_Toc165476428)

[**3.2.1 Архитектура** 41](#_Toc165476429)

[**3.2.2 Модуль блокчейна** 46](#_Toc165476430)

[**3.2.3 Модуль ноды** 55](#_Toc165476431)

# **Определения, обозначения и сокращения**

В настоящей работе применены сокращения и термины с соответствующими определениями.

Блокчейн (с англ. block chain –цепь блоков)  — выстроенная по определённым правилам непрерывная последовательная цепочка блоков (связный список), содержащих какую-либо информацию. Связь между блоками обеспечивается не только нумерацией, но и тем, что каждый блок содержит свою собственную [хеш-сумму](https://ru.wikipedia.org/wiki/%D0%A5%D0%B5%D1%88-%D1%81%D1%83%D0%BC%D0%BC%D0%B0) и хеш-сумму предыдущего блока.

Proof-of-Authority (PoA) (с англ. Доказательство полномочий) — это алгоритм консенсуса, подходящий для частного и разрешенного блокчейна, который опирается на ряд заранее выбранных авторитетных узлов, называемых валидаторами.

Proof-of-Stake (PoS) (с англ. Доказательство владения) — это класс механизмов консенсуса для блокчейнов, которые работают путем выбора валидаторов пропорционально их количеству активов в соответствующей криптовалюте. Это сделано для того, чтобы избежать вычислительных затрат на схемы доказательства работы (POW).

Криптоактивы — это цифровые представления стоимости, основанные на криптографии и децентрализованной одноранговой архитектуре, основанной на технологии распределенного реестра (DLT), которая позволяет двум сторонам напрямую взаимодействовать друг с другом без необходимости в доверенном посреднике.

Фреймворк (с англ. framework – «каркас, структура») – заготовка, готовая модель в IT для быстрой разработки, на основе которой можно дописать собственный код. Он задает структуру, определяет правила и предоставляет необходимый набор инструментов для создания проекта. [1]

ИС – Информационная система.

СУБД (Система управления базами данных) – это инструментальная оболочка пользователя, а ввиду того, что такая среда ориентирована на немедленное удовлетворение запросов пользователя – это всегда система интерпретатор. [2]

HTTP (протокол передачи гипертекста) – это протокол прикладного уровня для Всемирной паутины, составляющий ее самое сердце. HTTP реализуется в двух частях приложений: клиентской и серверной. Клиентская и серверная части программ, исполняемые на различных конечных системах, общаются друг с другом, обмениваясь сообщениями HTTP. Протокол HTTP определяет структуру этих сообщений и порядок обмена между клиентом и сервером. [3]

API (англ. Application Programming Interface – программный интерфейс приложения) – это набор способов и правил, по которым различные программы общаются между собой и обмениваются данными. [4]

SQL (Structured Query Language – язык структурированных запросов) – это язык программирования, который используется для работы с базами данных. [5]

PIP – система управления пакетами, которая используется для установки и управления программными пакетами, написанными на Python.

Форк в блокчейне - это разветвление или разделение цепи блоков на две или более отдельные цепи из-за несогласия в сети по правилам протокола. Это может произойти из-за различий в мнениях участников сети относительно обновлений протокола, конфликтов в консенсусе или атак на сеть. Каждая ветвь может продолжать свое существование, и участники сети должны выбрать, какую ветвь поддерживать.

JSON (JavaScript Object Notation) – это легковесный синтаксис для обмена данными, которые понимают как люди, так и машины. Как следует из названия, JSON основан на скриптовом языке JavaScript; однако, JSON сам по себе полностью не зависим от языка. [6]

Python – мощный компьютерный язык программирования, поддерживающий множество парадигм, который оптимизирован для обеспечения высокой продуктивности программистов, читабельности кода и качества программного обеспечения. [7]

Хеш - результат применения хэш-функции к набору данных, который преобразует входные данные произвольной длины в строку фиксированной длины. Этот хеш является уникальным и практически невозможно восстановить оригинальные данные из него. Хеши используются в криптовалюте для различных целей, таких как создание уникальных идентификаторов для блоков и транзакций, проверка целостности данных, а также обеспечение безопасности и подтверждение подлинности информации. [8]

Дерево Меркла - это структура данных, используемая в криптографии и блокчейне для эффективной проверки целостности и подтверждения содержимого больших объемов данных. Оно строится путем последовательного хэширования пары узлов данных (обычно транзакций) до тех пор, пока не будет получено единое значение, называемое корневым хешем. Этот корневой хеш затем используется для проверки целостности всего дерева: если корневой хеш совпадает, значит, все данные в дереве неизменны и целостны. Деревья Меркла позволяют быстро и эффективно проверять целостность больших объемов данных, что делает их важным элементом в технологиях блокчейна и криптографии.[9]

# **Введение**

В наше время, в эру быстрого развития цифровых технологий и финансовых инноваций, возникает все более актуальная потребность в создании эффективных механизмов для проведения трансграничных финансовых операций между различными цифровыми валютами. Для удовлетворения этой потребности становится необходимым разработать инновационные решения, которые сочетают в себе высокий уровень безопасности, прозрачности и эффективности.

В рамках данной дипломной работы будет представлен механизм прямых трансграничных финансовых операций между цифровыми валютами различных стран на основе технологии блокчейн с использованием протокола Proof of Authority (PoA). Этот механизм представляет собой инновационное решение, способное решить множество проблем, с которыми сталкиваются участники международных финансовых транзакций.

Перед тем как погрузиться в детали разработки механизма, важно рассмотреть некоторые аспекты предыстории. С появлением цифровых валют, таких как Bitcoin, Ethereum и другие, возникла потребность в их использовании для международных финансовых операций. Однако, существующие методы часто ограничены проблемами безопасности, временем подтверждения транзакций и высокими комиссиями.

Одним из способов решения этих проблем является применение технологии блокчейн на основе протокола Proof of Authority (PoA). Этот подход обеспечивает высокую степень безопасности и эффективности за счет использования ограниченного числа доверенных участников для подтверждения транзакций.

Целью данной работы является разработка и реализация механизма прямых трансграничных финансовых операций, который будет обладать высокой степенью надежности, прозрачности и эффективности благодаря использованию технологии блокчейн на основе протокола Proof of Authority. Работа будет включать в себя анализ требований, проектирование системы, ее реализацию и тестирование с целью обеспечения полной функциональности и соответствия поставленным целям.

В итоге, успешная реализация данного механизма будет способствовать развитию международных финансовых отношений, упрощению процессов проведения трансграничных операций и повышению уровня доверия участников рынка к цифровым валютам как инструменту международных расчетов.

Для реализации системы прямых трансграничных финансовых операций между цифровыми валютами различных стран на базе технологии блокчейн с использованием протокола Proof of Authority (PoA) необходимо выполнить следующие основные задачи:

1. Анализ требований: Определение функциональных и нефункциональных требований к системе, включая необходимость прямых трансграничных финансовых операций, безопасность, скорость подтверждения транзакций, совместимость с различными цифровыми валютами и соответствие регулятивным требованиям.
2. Проектирование системы: Разработка архитектуры системы, включая выбор технологий блокчейн, определение структуры сети с особым вниманием к синхронизации нод, а также определение интерфейсов для взаимодействия с системой.
3. Реализация особенной синхронизации нод: Разработка механизмов синхронизации нод с возможностью изменения кода ноды и асинхронной работы, включая создание механизмов для управления собственной веткой блоков.
4. Реализация веб-интерфейса и API: Создание пользовательского веб-интерфейса для управления аккаунтами, просмотра истории транзакций, а также реализация API для интеграции с внешними приложениями и системами.
5. Реализация технологии генерации адресов транзакций: Разработка механизмов для генерации уникальных адресов для проведения транзакций, обеспечивая безопасность и уникальность каждого адреса.
6. Развертывание блокчейн сети: Создание и настройка блокчейн сети на базе протокола Proof of Authority (PoA), включая выбор участников сети (доверенных нод) и установку соответствующих параметров сети.
7. Интеграция с внешними системами: Разработка и настройка интерфейсов для взаимодействия с внешними системами, такими как цифровые кошельки, биржи криптовалют и финансовые институты.
8. Тестирование и оптимизация: Проведение тестирования системы на соответствие требованиям, выявление и исправление ошибок, а также оптимизация производительности и безопасности.
9. Запуск и поддержка: Запуск системы в реальной среде, обеспечение непрерывной работы и поддержки, включая мониторинг производительности, обновление программного обеспечения и реагирование на возникающие проблемы.

Выполнение этих задач с учетом специфики цифровых валют и требований к международным финансовым операциям позволит создать эффективную и безопасную систему для проведения прямых трансграничных финансовых операций на основе технологии блокчейн.

# **Глава 1. Анализ предметной области**

## **Изучение существующих аналогов блокчейнов**

При изучении существующих аналогов блокчейнов и их механизмов консенсуса можно выделить несколько основных аспектов. Рассмотрим некоторые из них:

Bitcoin (Proof of Work):

Механизм консенсуса: Proof of Work (PoW) требует от майнеров выполнения вычислительно сложных задач для создания новых блоков и подтверждения транзакций.

Минусы: Высокая энергозатратность, медленная скорость подтверждения транзакций, возможность централизации майнинга в руках крупных игроков.

Ethereum (Proof of Stake в переходе):

Механизм консенсуса: В настоящее время Ethereum использует Proof of Work, но планирует перейти на Proof of Stake (PoS), где участники сети подтверждают транзакции на основе доли их криптовалюты в системе.

Минусы: Пока Ethereum использует PoW, он сталкивается с теми же проблемами, что и Bitcoin. В перспективе, PoS может столкнуться с проблемами связанными с концентрацией больших объемов криптовалюты в руках небольшого числа участников.

Ripple (Consensus Protocol):

Механизм консенсуса: Ripple использует собственный протокол Consensus, который позволяет участникам сети достигать согласия по порядку транзакций без выполнения вычислительных задач.

Минусы: Система требует доверия к центральной организации, контролирующей список узлов, участвующих в процессе консенсуса.

Теперь давайте рассмотрим, почему Proof of Authority (PoA) является наиболее подходящим механизмом консенсуса для системы трансграничных транзакций:

Эффективность и скорость: PoA обеспечивает быстрое время подтверждения транзакций за счет участия доверенных нод, что особенно важно для операций в реальном времени.

Низкая стоимость транзакций: PoA не требует вычислительных ресурсов, что позволяет снизить стоимость проведения транзакций по сравнению с PoW и PoS.

Высокий уровень безопасности: При правильной настройке списка доверенных нод PoA обеспечивает высокий уровень безопасности, защищая систему от атак.

Гибкость и управляемость: PoA позволяет управлять списком доверенных нод, что обеспечивает гибкость в настройке сети в соответствии с требованиями системы трансграничных транзакций.

Помимо этого, стоит отметить, что блокчейн на механизме консенсуса Proof of Authority (PoA) не требует nonce (случайного числа), который используется в механизме Proof of Work (PoW) для создания нового блока. Также PoA не требует от валидаторов выполнения бесполезной работы, что позволяет снизить энергозатраты и повысить эффективность сети.

Важно отметить, что механизм Proof of Work (PoW) подвержен риску взлома в случае, если злоумышленнику удастся контролировать более 51% вычислительной мощности сети или использовать другие методы атаки. В отличие от этого, разработанный блокчейн с механизмом Proof of Authority (PoA), как описано выше, не подвержен такому риску. Даже в случае попытки взлома одной из нод, произойдет форк сети, и ноды автоматически рассинхронизируются, предотвращая возможность взлома извне.

Таким образом, Proof of Authority представляет собой оптимальное решение для системы трансграничных транзакций, обеспечивая высокую скорость, низкую стоимость, высокий уровень безопасности и гибкость управления.

## **1.2 Анализ потребностей и проблем, которые приложение должно решать**

Это приложение должно решать несколько важных проблем и удовлетворять потребности в области международных финансовых операций. Одной из ключевых проблем, которую оно должно решать, является неэффективность и затратность традиционных методов международных платежей. Существующие системы часто подвержены высоким комиссиям и длительным срокам выполнения транзакций, что затрудняет быстрые и экономически эффективные международные финансовые операции.

Кроме того, существует проблема недостаточной прозрачности и безопасности при выполнении международных транзакций. В традиционных финансовых системах часто возникают вопросы относительно прозрачности операций и безопасности данных. Это создает недоверие и риск для участников операций.

Другой важной проблемой является сложность и медлительность процесса соблюдения регуляторных требований при проведении международных транзакций. Традиционные финансовые системы требуют многочисленных проверок и процедур для соблюдения регуляторных норм, что может замедлить процесс и увеличить издержки.

Таким образом, данное приложение должно решать проблемы высоких комиссий и задержек в международных платежах, обеспечивать прозрачность и безопасность операций, а также упрощать процесс соблюдения регуляторных требований.

Дополнительной потребностью, которую приложение должно решать, является обеспечение доверия к узлам блокчейна при отсутствии механизма доказательства работы (Proof of Work). В случае, когда в блокчейне присутствуют несколько узлов без механизма доказательства работы, становится сложно доверять каждой из них.

Для решения этой проблемы планируется создание системы, в которой владелец каждой ноды имеет возможность изменить свой код, в результате чего нода автоматически перестает синхронизироваться с другими узлами, а остальные узлы перестают с ней синхронизироваться. Благодаря этой возможности пользователь сможет самостоятельно выбирать узлы, которым он доверяет. Таким образом, реализуется автоматизированный легковесный форк, который позволяет пользователям контролировать собственную сеть и выбирать наиболее надежные узлы для проведения транзакций.

## **1.3 Выбор среды разработки**

При выборе языка программирования был выбран Python. Я рассмотрю причины, почему принял такое решение.

1. Простота и читаемость кода: Python известен своей простотой и понятностью синтаксиса, что делает его отличным выбором для разработки сложных систем, таких как блокчейн. Читаемый код упрощает поддержку и модификацию проекта.

2. Большое количество библиотек и фреймворков: Python обладает обширной экосистемой библиотек и фреймворков, которые могут значительно ускорить разработку блокчейн-приложений и веб-интерфейсов к ним. Например, для работы с блокчейном в Python используются библиотеки как "pyethereum", "web3.py", "pycryptodome" и многие другие.

3. Универсальность: Python поддерживается на множестве платформ и операционных систем, что обеспечивает универсальность при разработке блокчейна и веб-приложений для него. Разработчики могут создавать приложения для различных устройств и операционных систем, используя Python.

4. Поддержка многопоточности и асинхронности: Python поддерживает работу с многопоточностью, что может быть полезно при разработке высокопроизводительных приложений, таких как блокчейн. Также существует множество асинхронных библиотек для Python, позволяющих эффективно управлять асинхронными операциями.

5. Обширное сообщество разработчиков: Python имеет огромное и активное сообщество разработчиков, что означает наличие огромного количества ресурсов, обучающих материалов, форумов поддержки и др., что значительно облегчает процесс разработки и решение возможных проблем.

В совокупности эти факторы делают Python отличным выбором для разработки блокчейна и соответствующих веб-приложений, предоставляя разработчикам мощный инструментарий для создания надежных и производительных приложений.

В данном сравнительном анализе будут рассмотрены два фреймворка FastAPI и Flask, а также будут проанализированы их возможности, производительность, гибкость, расширяемость, и поддержка сообщества.

1. Производительность:

FastAPI обладает высокой производительностью за счет использования современных технологий, таких как Python 3.6+ и асинхронные запросы. Flask, в свою очередь, является более традиционным фреймворком и имеет менее эффективный механизм обработки запросов. FastAPI предлагает более быстрые скорости выполнения и обработки запросов.

1. Типизация данных и валидация:

FastAPI обеспечивает автоматическую валидацию запросов на основе типов данных, определенных в Python, что помогает предотвращать ошибки и улучшает безопасность приложения. Flask, в свою очередь, не предоставляет такой же уровень типизации и валидации данных.

1. Документация:

FastAPI автоматически генерирует документацию API на основе аннотаций Python, что делает процесс документирования API более простым и удобным. В Flask требуется дополнительное время и усилия для создания подробной документации.

1. Поддержка асинхронности:

FastAPI хорошо поддерживает асинхонное программирование и асинхронные запросы, что улучшает производительность приложения и позволяет эффективно обрабатывать большие объемы данных. Flask не так хорошо подходит для работы с асинхронными запросами.

1. Совместимость с OpenAPI и Swagger:

FastAPI построен на базе стандартов OpenAPI и Swagger, что облегчает интеграцию с другими сервисами и инструментами, поддерживающими эти стандарты. Flask не обладает такой же сильной поддержкой OpenAPI и требует дополнительных усилий для работы с ними.

В целом, FastAPI предлагает более современный подход к разработке веб-приложений, обеспечивая высокую производительность, автоматическую валидацию данных, удобную документацию и поддержку асинхронного программирования. Для блокчейна критерии асинхронности и скорости работы — это ключевые параметры для выбора фреймворка. Исходя из этого был выбран именно этот фреймворк.

## **1.4 Изучение возможностей фреймворка FastAPI**

FastAPI — это современный и эффективный веб-фреймворк на языке Python, который предоставляет разработчикам мощные инструменты для создания веб-приложений. Он позволяет быстро и легко разрабатывать API, обеспечивая высокую производительность за счет использования современных технологий.

Для разработки веб интерфейса к блокчнейну FastAPI может быть отличным выбором, так как он обладает рядом полезных возможностей. Например, благодаря асинхронной поддержке можно создать быстродействующие веб-приложение, способное обрабатывать большое количество запросов одновременно. Кроме того, FastAPI предоставляет интуитивный и простой в использовании интерфейс для описания схем данных API с помощью Pydantic, что облегчает разработку и поддержку кода.

Еще одним преимуществом FastAPI является возможность автоматической генерации документации API на основе аннотаций в коде, что значительно упрощает процесс документирования вашего приложения. Кроме того, FastAPI обладает встроенной поддержкой валидации данных, автоматической сериализации и десериализации запросов и многими другими полезными функциями.

Использование FastAPI для блокчейн-приложения позволит не только создать эффективное и надежное веб-приложение, но и изучить современные подходы к разработке API, работе с асинхронным кодом и использованию современных технологий веб-разработки.

### **1.4.1 Преимущества фреймворка FastAPI**

Одним из ключевых достоинств FastAPI является его высокая производительность. Благодаря применению современных технологий асинхронного программирования, данный фреймворк обеспечивает высокую скорость работы и эффективное использование ресурсов, что делает его отличным выбором для создания высоконагруженных веб-приложений.

Кроме того, FastAPI предлагает удобный и интуитивно понятный интерфейс для разработчиков. Он использует аннотации Python для описания эндпоинтов и схем данных API, что делает процесс создания веб-приложений более простым и эффективным. Благодаря автоматической генерации интерактивной документации на основе аннотаций в коде, разработчики могут быстро и легко создавать, и поддерживать документацию без дополнительных усилий.

Еще одним преимуществом FastAPI является встроенная поддержка валидации данных, что обеспечивает безопасность API и помогает избежать ошибок ввода. Благодаря поддержке различных форматов данных, таких как JSON, формы HTML, файлы и другие, FastAPI является универсальным инструментом для разработки разнообразных веб-приложений, обеспечивая разработчикам широкие возможности при создании функциональных и эффективных веб-сервисов.

### **1.4.2 Недостатки фреймворка FastAPI**

FastAPI, несмотря на свои многочисленные достоинства, также имеет некоторые недостатки. В частности, такой фреймворк может показаться избыточным для небольших проектов или приложений, требующих минималистического подхода. Кроме того, изучение особенностей и функционала FastAPI может потребовать времени и усилий, особенно для начинающих программистов, не знакомых с асинхронным программированием или аннотациями в Python.

Еще одним недостатком FastAPI является отсутствие широкого сообщества и большого количества сторонних библиотек и плагинов, как у более узнаваемых и распространенных фреймворков. Это может затруднить поддержку и развитие проекта в долгосрочной перспективе, особенно если возникнет необходимость в специфической функциональности, которая не предусмотрена по умолчанию в FastAPI.

Кроме того, из-за своей особенности использования асинхронного программирования FastAPI может создавать некоторые сложности в проектировании архитектуры приложения. Это может потребовать дополнительных усилий и времени для овладения соответствующими концепциями и методиками разработки, что также считается недостатком данного фреймворка.

## **1.5 Выводы по главе 1**

Я рассмотрел различные аспекты разработки блокчейна на основе механизма Proof of Authority (PoA) для системы трансграничных транзакций. Проведенный анализ выявил, что такой блокчейн способен эффективно решать ряд ключевых проблем, среди которых - высокие комиссии и задержки в международных платежах, неэффективность традиционных финансовых систем, недостаточная прозрачность и безопасность операций, а также сложности соблюдения регуляторных требований.

Изучение существующих аналогов блокчейнов позволило выявить различные механизмы консенсуса, такие как Proof of Work (PoW), Proof of Stake (PoS) и Consensus Protocol. Однако, по результатам анализа, было установлено, что механизм PoA представляет собой наиболее подходящий вариант для решения задач системы трансграничных транзакций, благодаря своей эффективности, низкой стоимости проведения транзакций, высокому уровню безопасности и гибкости управления.

Кроме того, в рамках анализа были выявлены дополнительные особенности предлагаемой системы, такие как возможность изменения кода узлов и автоматическая рассинхронизация сети в случае попытки взлома, что дополнительно повышает ее безопасность и надежность.

Таким образом, на основании проведенного анализа можно сделать вывод о целесообразности и перспективности использования блокчейна на основе механизма PoA для реализации системы трансграничных финансовых операций.

# **Глава 2. Проектирование приложения**

## **2.1 Определение требований к приложению**

Определение требований к приложению блокчейна на основе механизма Proof of Authority (PoA) для проведения трансграничных транзакций включает в себя ряд ключевых аспектов, которые необходимо учесть для успешной разработки и функционирования системы. Эти требования включают в себя:

1. Эффективность трансграничных платежей: Приложение должно обеспечивать быструю и надежную передачу средств между участниками из разных стран с минимальными комиссиями и задержками.
2. Прозрачность и безопасность операций: Система должна обеспечивать высокий уровень прозрачности и безопасности всех транзакций, чтобы участники могли доверять целостности и результатам проведенных операций.
3. Соответствие регуляторным требованиям: Приложение должно быть способно соблюдать регуляторные нормы и требования, действующие в различных странах, чтобы предотвратить возможные конфликты и проблемы с законодательством.
4. Гибкость и масштабируемость: Система должна быть гибкой и легко масштабируемой, чтобы адаптироваться к изменяющимся потребностям рынка и обеспечивать возможность обработки большого объема транзакций.
5. Управление доступом и безопасность: Приложение должно обеспечивать строгий контроль доступа к системе и конфиденциальность информации, защищая данные пользователей от несанкционированного доступа и взлома.
6. Легкость использования и интеграции: Система должна быть удобной в использовании как для конечных пользователей, так и для предприятий, а также легко интегрироваться с существующими финансовыми и технологическими системами.
7. Автоматизация процессов: Приложение должно обеспечивать автоматизацию ключевых процессов, связанных с проведением трансграничных транзакций, чтобы уменьшить ручную работу и повысить эффективность операций.

Учитывая эти требования, разработка и реализация приложения блокчейна на основе механизма PoA должны быть направлены на обеспечение высокого качества обслуживания и удовлетворение потребностей всех заинтересованных сторон.

## **2.2 Разработка структуры приложения и его компонентов**

Основные компоненты приложения и их взаимосвязи:

1. Chain:

- Описание: Этот компонент содержит базу данных блоков и транзакций. Здесь осуществляется генерация мнемоник, создание и проверка подписей, а также получение и создание транзакций и блоков.

- Взаимосвязь с другими компонентами: Chain взаимодействует с компонентом Crypto для шифрования мнемонических фраз и выполнения математических операций, необходимых для создания и проверки подписей. Также Chain обеспечивает данные, которые могут быть использованы Node для проверки и добавления новых блоков в цепочку.

2. Crypto:

- Описание: Этот модуль отвечает за создание транзакций, составление транзакций и компоновку байтов, а также за шифрование мнемонических фраз и выполнение математических операций.

- Взаимосвязь с другими компонентами: Crypto используется Chain для создания и проверки подписей транзакций, а также для шифрования мнемонических фраз. Кроме того, он может взаимодействовать с Node для создания новых транзакций и блоков.

3. Node:

- Описание: Этот модуль представляет собой ноду блокчейна, которая выпускает и валидирует блоки. У ноды есть свой API, который используется другими компонентами приложения.

- Взаимосвязь с другими компонентами: Node использует данные из Chain для проверки целостности блоков и транзакций, а также может использовать Crypto для создания новых блоков. Его API доступен компоненту Web для взаимодействия с пользователем.

4. Web (Веб-интерфейс):

- Описание: Этот модуль предоставляет графический интерфейс пользователям через веб-сайт. Включает в себя HTML и JS скрипты для взаимодействия с пользователем.

- Взаимосвязь с другими компонентами: Web взаимодействует с Node через его API для получения данных о блоках и транзакциях, а также для отправки новых транзакций.

Каждый из этих компонентов играет ключевую роль в функционировании приложения, а их взаимосвязь обеспечивает согласованную работу системы в целом.

Chain служит основным хранилищем данных блокчейна, предоставляя информацию о блоках и транзакциях для других компонентов. Crypto обеспечивает безопасность и целостность данных благодаря криптографическим методам, а также осуществляет создание и проверку транзакций. Node является ядром сети, отвечая за создание новых блоков и проверку их на валидность, а также предоставляет API для взаимодействия с другими компонентами. Web обеспечивает пользовательский интерфейс, позволяя пользователям взаимодействовать с блокчейном через веб-интерфейс.

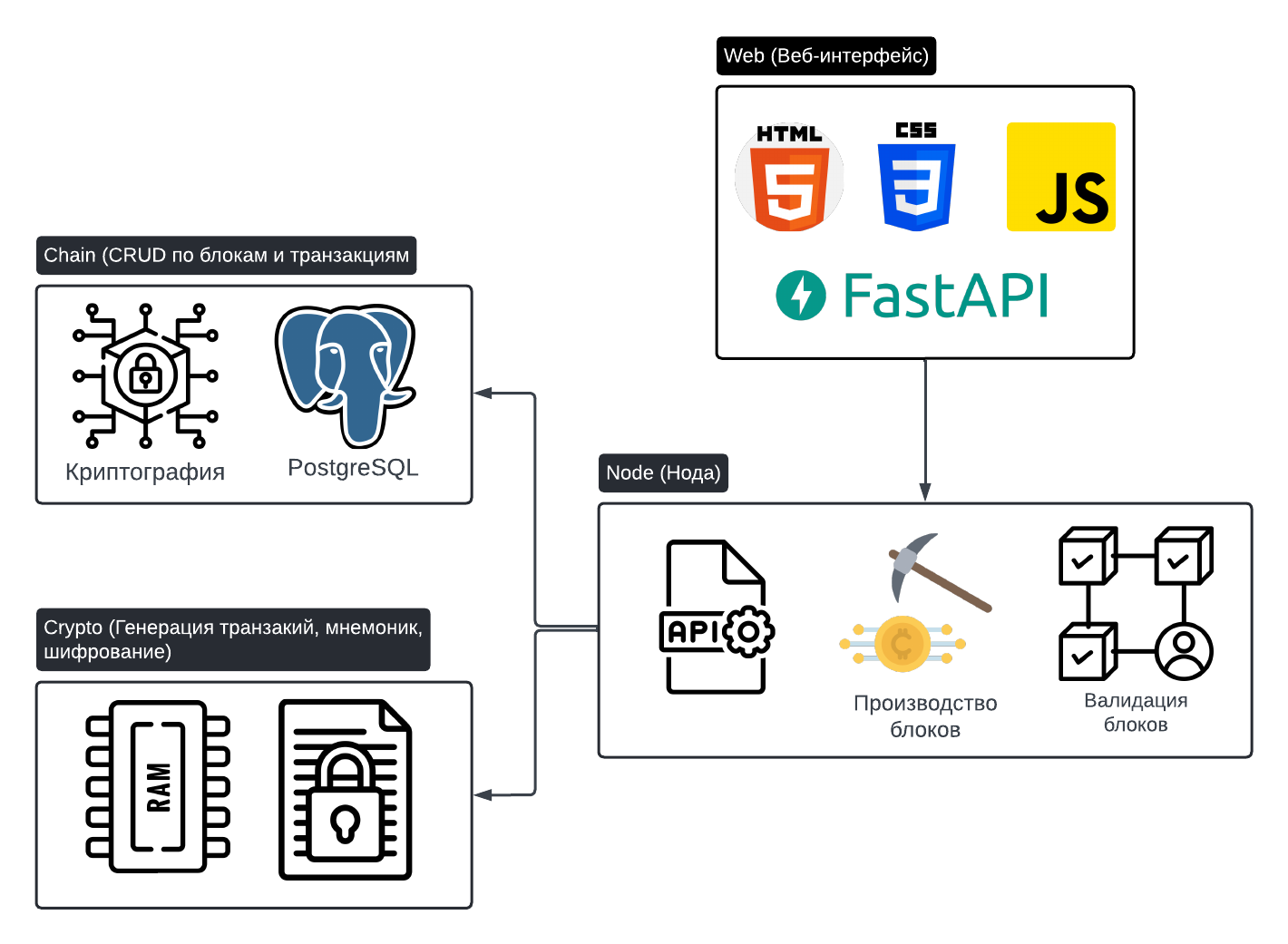


Рисунок 1 – архитектура модулей

Таким образом, компоненты взаимодействуют между собой, обмениваясь данными и обеспечивая работоспособность и эффективность всей системы блокчейна на механизме PoA для проведения трансграничных транзакций.

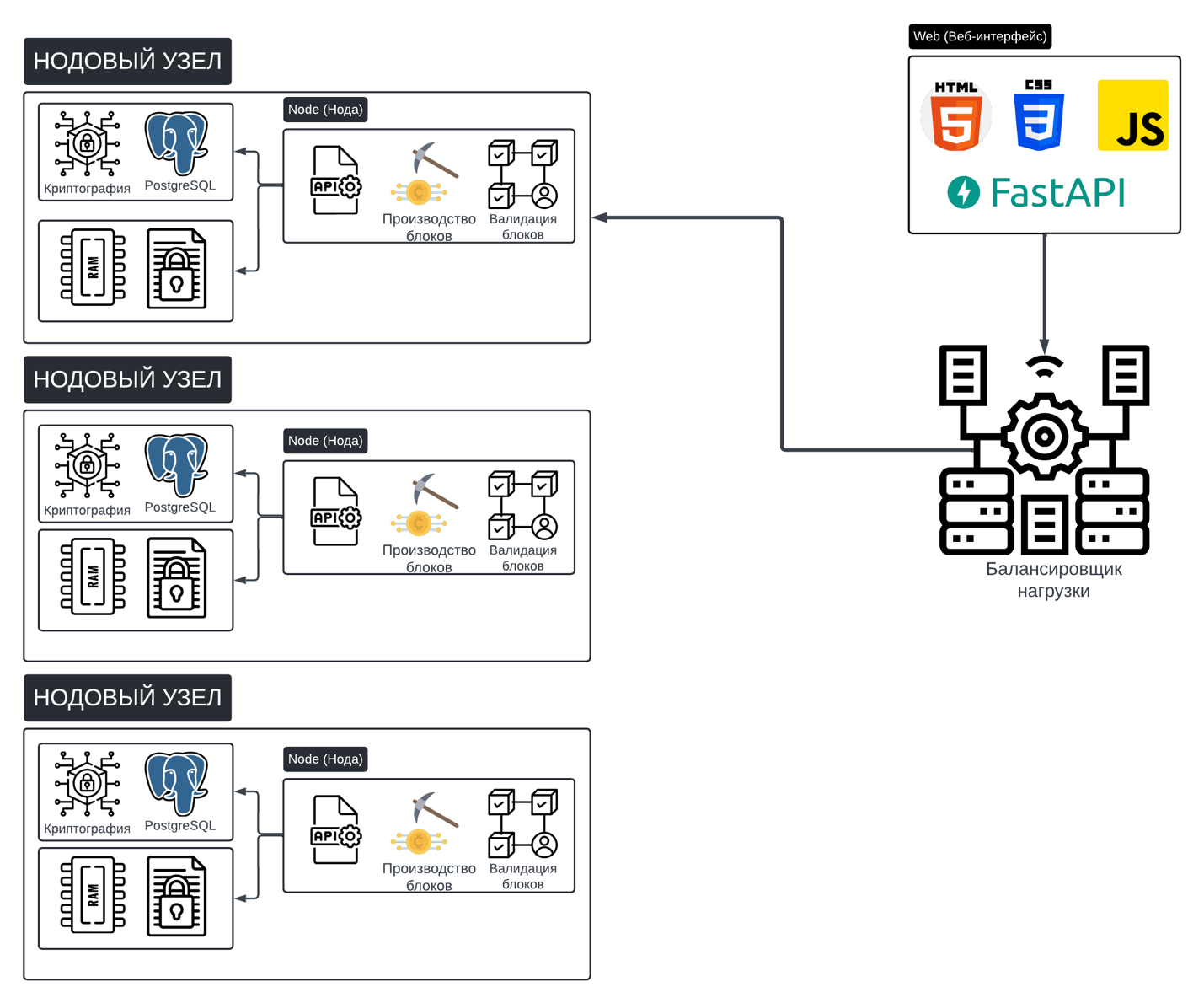


Рисунок 2 – архитектура модулей с балансировщиком

## **2.3 Разработка интерфейса пользователя**

В данном разделе рассмотрена разработка интерфейса пользователя для веб версии блокчейна на PoA.

Для разработки интерфейса пользователя блокчейн-приложения с учетом предложенных компонентов следует уделить особое внимание каждому из них.

1. Окно аутентификации:

- Форма ввода мнемоники для аутентификации пользователя.

- Возможность генерации мнемоники для нового пользователя.

2. Кошелек:

- Отображение баланса пользователя и его адрес в удобном для восприятия виде.

- Вкладки для отправки криптовалюты и создания подписей, где пользователь сможет вводить необходимые данные для совершения операции.

- Вкладки для просмотра транзакций, где отображаются все отправленные и полученные транзакции пользователя с возможностью сортировки и фильтрации.

3. Окно просмотра транзакций:

- Отображение детальной информации о каждой транзакции, включая ее характеристики и данные блока, в котором она была включена.

- Возможнлсть просмотра связанных транзакций для лучшего понимания контекста операции.

4. Окно просмотра блока:

- Отображение всех данных блока, включая список транзакций, включенных в него.

- Возможность просмотра подробной информации о каждой транзакции в блоке.

5. Окно обзорщика блоков:

- Визуализированный процесс генерации блоков в реальном времени.

- Поиск конкретного блока по его номеру или транзакции по ее хешу для быстрого доступа к нужной информации.

Помимо этого, обеспена плавная навигация между различными окнами и возможность быстрого доступа к основной функциональности приложения. Уделено внимание дизайну интерфейса, он сделан его интуитивно понятным и привлекательным для пользователя.

## **2.4 Описание архитектуры приложения**

Архитектура приложения для системы трансграничных транзакций в виде блокчейна на механизме Proof of Authority (PoA) включает в себя следующие компоненты:

1. Клиентский веб-интерфейс:

- Веб-приложение, которое предоставляет пользовательский интерфейс для взаимодействия с блокчейном.

- Реализует функциональность аутентификации пользователей, просмотра баланса и истории транзакций, отправки транзакций и просмотра информации о блоках.

- Использует HTML, CSS и JavaScript для создания пользовательского интерфейса и взаимодействия с API нод.

2. Субд Postgresql:

- Система управления базами данных, используемая для хранения информации о блоках, транзакциях и других данных блокчейна.

- Хранит информацию о пользовательских аккаунтах, балансах, истории транзакций и другие сведения, необходимые для функционирования приложения.

3. Ноды:

- Узлы блокчейна, отвечающие за создание, проверку и хранение блоков.

- Каждая нода в сети обрабатывает транзакции, создает новые блоки и участвует в процессе достижения консенсуса.

- Ноды взаимодействуют между собой для распространения блоков и поддержания целостности сети.

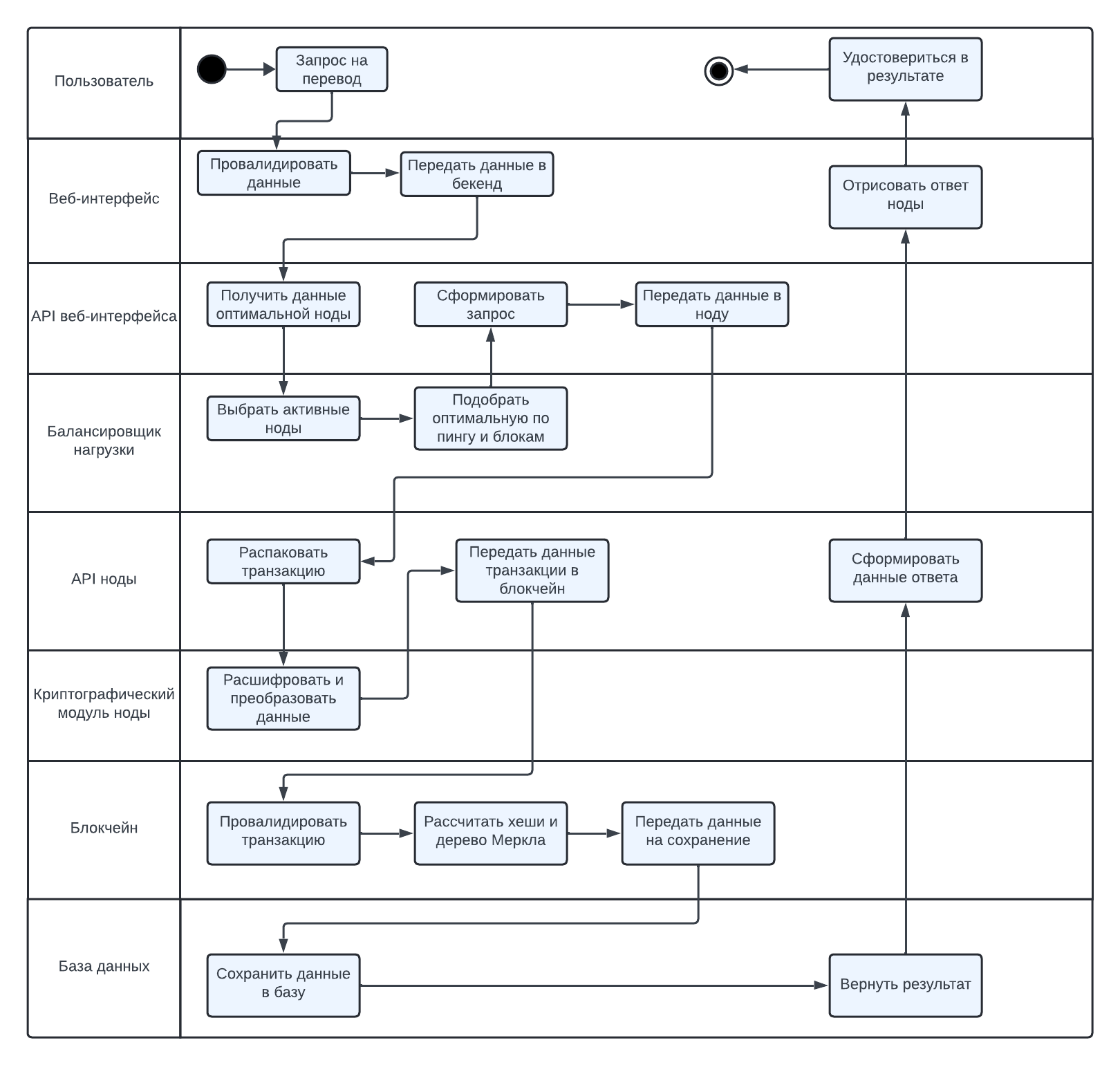


Рисунок 3 – Путь запроса по архитектуре

Архитектура приложения предполагает, что клиентский веб-интерфейс взаимодействует с нодами блокчейна через их API для получения информации о блоках, транзакциях и выполнения операций с аккаунтами. Ноды в свою очередь взаимодействуют друг с другом для распространения блоков и поддержания актуального состояния блокчейна. Данные о блоках, транзакциях и аккаунтах хранятся в базе данных Postgresql для обеспечения удобного доступа и быстрой обработки информации.

## **2.5 Выводы по главе 2**

Анализируя вышеописанные модули и архитектуру приложения, можно сделать следующие выводы.

Клиентский веб-интерфейс играет ключевую роль в обеспечении пользовательского опыта. Он предоставляет удобный доступ к функционалу блокчейна, включая просмотр информации о транзакциях и блоках, а также выполнение операций с аккаунтами.

Использование системы управления базами данных Postgresql обеспечивает надежное хранение данных блокчейна и обеспечивает быстрый доступ к ним. Это важно для обеспечения эффективной работы приложения и обеспечения целостности данных.

Ноды блокчейна играют ключевую роль в обеспечении безопасности и целостности сети. Механизм PoA обеспечивает высокую эффективность и надежность работы нод, что является важным аспектом для системы трансграничных транзакций.

В целом, архитектура приложения объединяет эти компоненты в единую систему, обеспечивая высокий уровень безопасности, прозрачности и эффективности для пользователей.

# **Глава 3. Разработка приложения**

## **3.1 Создание хранилища для блокчейна и его описание**

В данной главе проводем анализ различных видов хранилищ данных и их применимости для блокчейна на механизме Proof of Authority (PoA). В этой главе сравним следующие виды хранилищ:

1. Реляционные базы данных:

- Примеры: PostgreSQL, MySQL, Oracle.

- Плюсы: Обеспечивают структурированное хранение данных с возможностью использования сложных запросов. Обеспечивают надежность и целостность данных.

- Минусы: Могут быть менее масштабируемыми по сравнению с NoSQL базами данных. Требуют более тщательного проектирования схемы данных.

2. NoSQL базы данных

- Примеры: MongoDB, Cassandra, Redis.

- Плюсы: Обеспечивают гибкое хранение данных без жесткой схемы. Обладают высокой масштабируемостью и способностью обрабатывать большие объемы данных.

- Минусы: Могут быть менее эффективными при выполнении сложных запросов, требуют дополнительной работы по обеспечению согласованности данных.

3. Файловые системы:

- Примеры: Amazon S3, Google Cloud Storage, IPFS.

- Плюсы: Простота использования, низкая стоимость хранения данных. Поддерживают различные типы данных, включая файлы и документы.

- Минусы: Могут быть менее эффективными при выполнении операций поиска и обработки данных. Могут не обеспечивать такой же уровень надежности и безопасности, как реляционные базы данных.

PostgreSQL является мощной реляционной базой данных, которая обладает рядом характеристик, делающих ее подходящим выбором для использования в блокчейне на механизме Proof of Authority (PoA):

1. Структурированное хранение данных: PostgreSQL позволяет организовать данные в виде таблиц с жесткой схемой, что особенно полезно для блокчейна, где необходимо хранить информацию о блоках, транзакциях, аккаунтах и других аспектах сети.

2. Транзакционная безопасность: PostgreSQL обеспечивает ACID-свойства транзакций (Atomicity, Consistency, Isolation, Durability), что гарантирует целостность данных и надежность операций в блокчейне. Это особенно важно для обеспечения согласованности данных в децентрализованной среде.

3. Поддержка сложных запросов: PostgreSQL предоставляет широкие возможности для выполнения сложных запросов и аналитики данных. Это позволяет эффективно работать с большим объемом информации, характерным для блокчейна.

4. Масштабируемость: PostgreSQL обладает возможностью масштабироваться как вертикально (путем увеличения ресурсов сервера), так и горизонтально (путем использования репликации и шардинга). Это позволяет блокчейну на PoA обрабатывать большие объемы данных и поддерживать высокую производительность.

5. Большое сообщество и поддержка: PostgreSQL является одной из самых популярных открытых реляционных баз данных, что обеспечивает широкую поддержку, обновления и интеграцию с другими инструментами и технологиями.

6. Открытый исходный код: PostgreSQL распространяется под лицензией открытого исходного кода, что делает его доступным для использования без затрат на лицензирование и обеспечивает прозрачность разработки и независимость от поставщика.

Все эти факторы делают PostgreSQL привлекательным выбором для хранения данных в блокчейне на PoA, обеспечивая надежность, производительность и гибкость в работе с данными.

Перейдем к описанию структуры базы данных. База данных будет состоять из нескольких таблиц, каждая из которых будет хранить определенный тип данных. Рассмотрим каждую таблицу подробнее.

1. Таблица «block»

Эта таблица содержит в себе цепочку блоков. Состоит из следующих полей:

- id: Первичный уникальный ключ блока.

- block\_number: Числовой уникальный номер блока начиная с genesis блока

- block\_hash: Хеш блока, созданный на основе его атрибутов (номера блока, дерева меркла, предыдущего блока, времени итд) (текстовый формат)

- previous\_hash: Хеш предыдущего блока (текстовый формат)

- merkle\_root: Дерево меркла блока, построенное на основе последовательного хеширования транзакций блока, может быть null если транзакций в блоке нет (текстовый формат)

- authority\_id: Опознавательный ключ ноды валидатора которая выпустила данный блок (текстовый формат)

- timestamp: Дата выпуска блока в микросекундах

1. Таблица «transaction»

- id: Первичный уникальный ключ транзакции.

- sender\_address: Адрес отправителя транзакции (текстовый формат)

- recipient\_address: Адрес получателя транзакции (текстовый формат)

- amount: Числовая сумма отправленных активов. Хранится в базе в целочисленном виде, отображается с точностью 2 знака только визуально.

- timestamp: Дата совершения транзакции в микросекундах

- transaction\_hash: Хеш транзакции составленный на основе ее данных (отправителя, получателя, суммы, времени)

- block\_id: Ключ связанного блока в базе данных

- block\_number: Номер блока. После подтверждения транзакции она попадает в блок, а после подтверждения блока в это поле вписывается его номер.

После проектирования структуры базы данных для блокчейна необходимо приступить к созданию таблиц. Для этого используется язык SQL, который позволяет определить имена таблиц, их атрибуты и типы данных. Каждая таблица также должна иметь первичные ключи, индексы и ограничения в соответствии с требованиями. Этот процесс осуществляется с помощью SQL-инструкций CREATE TABLE.

После создания таблиц необходимо заполнить их данными. Для этого используется SQL-инструкция INSERT, которая позволяет добавить записи в таблицы. Например, можно добавить информацию о блоках и транзакциях, которые являются основными элементами блокчейна.

В результате выполнения всех этих шагов будет создана база данных, которая соответствует требованиям блокчейна. База данных будет содержать необходимые таблицы с соответствующей структурой, а также заполнена актуальными данными. Создание базы данных является важным этапом разработки, поскольку она является основой для работы приложения и обеспечивает эффективное хранение и управление данными.

Таким образом, создание структуры базы данных позволяет эффективно организовать хранение информации о блоках и транзакциях, что является основой для функционирования блокчейна.

**3.2 Реализации логики работы блокчейна**

В данном разделе представлен процесс разработки программного кода, который обеспечивает стабильное и эффективное функционирование блокчейна, нод, криптографических модулей и веб-интерфейса. Основные технические детали и ключевые аспекты работы кода детально рассмотрены с целью выявления основных принципов его функционирования.

Описание каждого модуля происходит в логическом порядке, начиная с архитектурного обзора и перехода к более детальному рассмотрению каждого компонента. Особое внимание уделяется технологии синхронизации нод и их консенсуса, а также возможности автоматического мягкого форка, что представляет собой уникальную особенность разрабатываемого решения.

Помимо этого, в работе происходит анализ плюсов и уникальных особенностей применяемой технологии, а также рассматриваются потенциальные пути для ее дальнейшего развития и усовершенствования. Все это призвано создать полное и глубокое понимание работы и возможностей разработанного программного продукта.

**3.2.1 Архитектура**

Для эффективной установки всех необходимых пакетов проекта следует создать файл "requirements.txt", перечислив в нем все библиотеки и их версии, а затем воспользоваться инструментом pip для их установки. Этот подход значительно упрощает процесс управления зависимостями, позволяя легко обновлять библиотеки и гарантировать совместимость среды разработки.

Создание базовой структуры проекта важно для сохранения порядка и удобства работы. Подобная организация каталогов по модулям и функциональности помогает разработчикам легко найти нужные файлы и облегчает сопровождение кода. Каждый модуль может содержать свои собственные файлы, отвечающие за конкретные аспекты проекта, что способствует его более эффективной разработке и поддержке.

Итоговая архитектура проекта в файловой системе:

.

├── alembic.ini

├── chain

│   ├── \_\_init\_\_.py

│   ├── block.py

│   ├── constants.py

│   ├── db

│   │   ├── \_\_init\_\_.py

│   │   ├── base.py

│   │   ├── block.py

│   │   ├── session.py

│   │   └── transaction.py

│   ├── timestamps.py

│   └── transaction.py

├── chain\_config.py

├── crypto

│   ├── \_\_init\_\_.py

│   ├── bech32.py

│   ├── converter.py

│   ├── generate\_wallet.py

│   ├── poa\_mnemonic.py

│   ├── sign.py

│   └── transfer.py

├── migration

│   ├── README

│   ├── env.py

│   ├── script.py.mako

│   └── versions

│   ├── 004766e0c634\_.py

│   ├── 7477f995b497\_initial.py

│   ├── 81396b4c6d04\_.py

│   ├── 8da8420576da\_.py

│   ├── 9f31208dc33e\_.py

│   ├── dd7cc63002ed\_.py

│   ├── ee3fd4166e49\_.py

│   └── efc133a3c757\_.py

├── node

│   ├── \_\_init\_\_.py

│   ├── api

│   │   ├── \_\_init\_\_.py

│   │   ├── block.py

│   │   ├── distributor.py

│   │   ├── status.py

│   │   ├── transaction.py

│   │   └── user.py

│   ├── app.py

│   ├── blockchain

│   │   ├── \_\_init\_\_.py

│   │   ├── balancer.py

│   │   ├── block\_processing.py

│   │   ├── startup.py

│   │   └── validate.py

│   ├── models

│   │   ├── \_\_init\_\_.py

│   │   ├── block.py

│   │   └── transaction.py

│   ├── routers

│   │   ├── \_\_init\_\_.py

│   │   ├── alive.py

│   │   ├── block.py

│   │   ├── receive\_block.py

│   │   ├── receive\_transaction.py

│   │   ├── transactions.py

│   │   └── user.py

│   ├── structs

│   │   ├── \_\_init\_\_.py

│   │   └── block.py

│   └── utils.py

├── node\_constants.py

├── requirements.txt

├── run\_node\_gaz.sh

├── run\_node\_sber.sh

├── run\_node\_yandex.sh

├── run\_web.sh

└── web

├── \_\_init\_\_.py

├── app.py

├── constants.py

├── encryption.py

├── file\_response.py

├── models

│   ├── \_\_init\_\_.py

│   └── mnemonic.py

├── routers

│   ├── \_\_init\_\_.py

│   ├── address\_watcher.py

│   ├── blocks\_watcher.py

│   ├── index.py

│   ├── mnemonic.py

│   ├── sign.py

│   ├── transaction.py

│   └── wallet.py

├── static

│   ├── css

│   │   ├── index.css

│   │   └── wallet.css

│   ├── html

│   │   ├── address.html

│   │   ├── block.html

│   │   ├── blocks.html

│   │   ├── login.html

│   │   ├── transaction.html

│   │   └── wallet.html

│   ├── images

│   │   ├── favicon.ico

│   │   ├── logback.jpg

│   │   └── logo.png

│   └── js

│   ├── block.js

│   ├── blocks.js

│   ├── index.js

│   ├── transaction.js

│   └── wallet.js

└── utils.py

**3.2.2 Модуль блокчейна**

Модуль блокчейна отвечает за хранение, создание, обновление и удаление блоков и транзакций. А также рассчет хешей на основе внутренних данных блоков и транзакций.

Блоки в блокчейне являются основными элементами данных, которые хранят информацию о транзакциях, контрактах или других событиях. Они связаны между собой в цепочку, обеспечивая непрерывность и безопасность хранения данных. Каждый блок содержит ссылку на предыдущий блок и хеш, который обеспечивает целостность и невозможность изменения прошлых записей. Блоки позволяют всем участникам системы видеть историю изменений и подтверждать их подлинность, обеспечивая надежность и безопасность работы блокчейна.

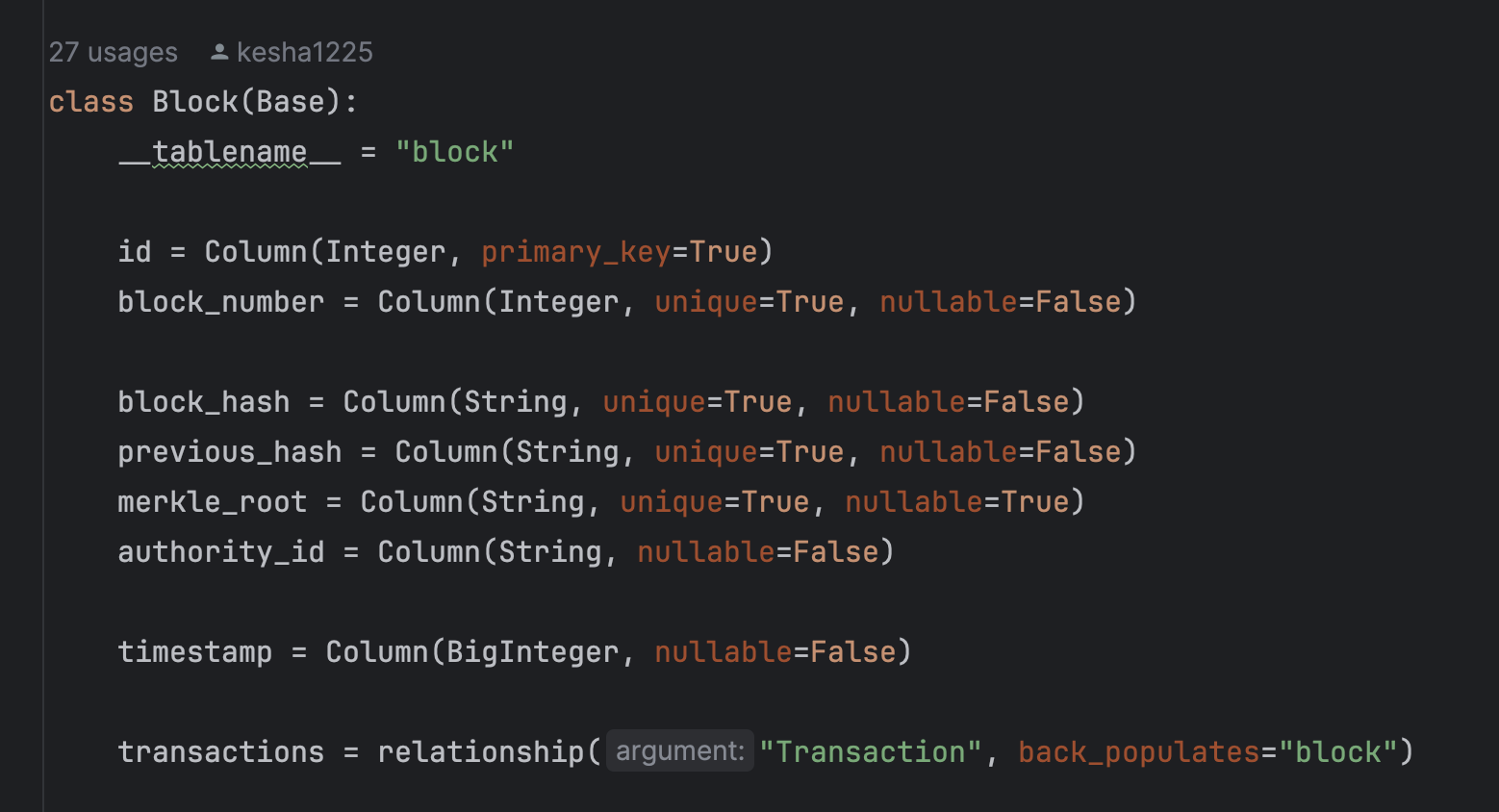


Рисунок 4 – структура блока в базе данных

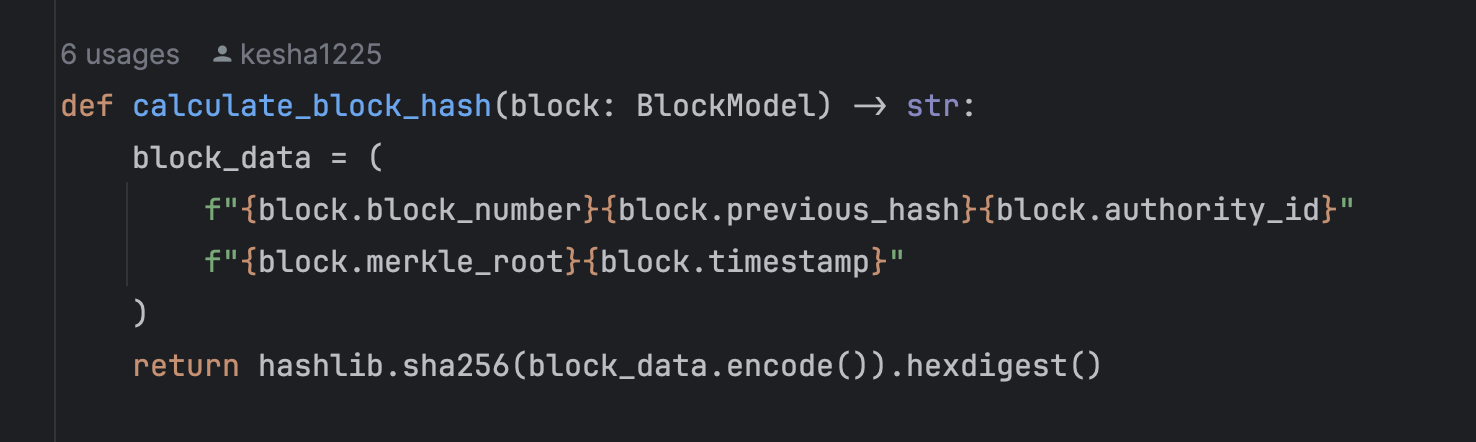


Рисунок 5 – функция рассчета хеша блока

Nonce (number used once) - это случайное число, используемое в процессе майнинга блоков в блокчейне Proof of Work (PoW). Он добавляется к блоку вместе с другими данными, чтобы изменить хеш блока и удовлетворить сложность доказательства работы (PoW). В блокчейне на PoW, майнеры должны перебирать различные значения nonce до тех пор, пока не найдут подходящий хеш блока.

В блокчейне на Proof of Authority (PoA) nonce не требуется, поскольку в этом типе консенсуса блоки создаются определенными авторизованными узлами, а не майнерами, конкурирующими за решение математической задачи. Необходимость в nonce отпадает, поскольку в PoA нет необходимости доказывать выполнение работы для создания блоков. Это позволяет упростить процесс формирования блоков и увеличить производительность сети, так как нет необходимости тратить ресурсы на вычисление подходящего nonce.

Отсутствие nonce в блокчейне на PoA можно считать важным улучшением, поскольку это уменьшает энергозатраты, связанные с процессом майнинга, и повышает эффективность сети блокчейна. Благодаря этому блокчейн на PoA становится более экологически чистым и масштабируемым, что важно для его долгосрочной устойчивости и развития.

Отсутствие лимита на количество транзакций в блокчейне является важным преимуществом и отличием от других блокчейнов, таких как биткоин и другие медленные криптовалюты. Вот несколько причин, почему это так:

Высокая пропускная способность: Блокчейн без ограничения на количество транзакций в блоке может обрабатывать больше транзакций за тот же период времени, что увеличивает общую пропускную способность сети. Это позволяет ускорить время подтверждения транзакций и обеспечить более быстрый оборот средств.

Отсутствие комиссий: Поскольку блоки могут вмещать больше транзакций, конкуренция за включение транзакций в блок становится менее острой. Это позволяет убрать комиссии за транзакции для пользователей, что делает использование блокчейна более доступным и экономически выгодным.

Улучшенная масштабируемость: Отсутствие ограничений на количество транзакций в блоке способствует улучшению масштабируемости блокчейна. Блокчейны с высокой пропускной способностью могут эффективно обслуживать большое количество пользователей и транзакций без существенного снижения производительности.

Улучшенный пользовательский опыт: Более быстрые и дешевые транзакции делают использование блокчейна более удобным для пользователей, улучшая общий пользовательский опыт и привлекательность платформы.

Таким образом, отсутствие лимита на количество транзакций в блоке способствует повышению эффективности и производительности блокчейна, что делает его более конкурентоспособным и привлекательным для пользователей.

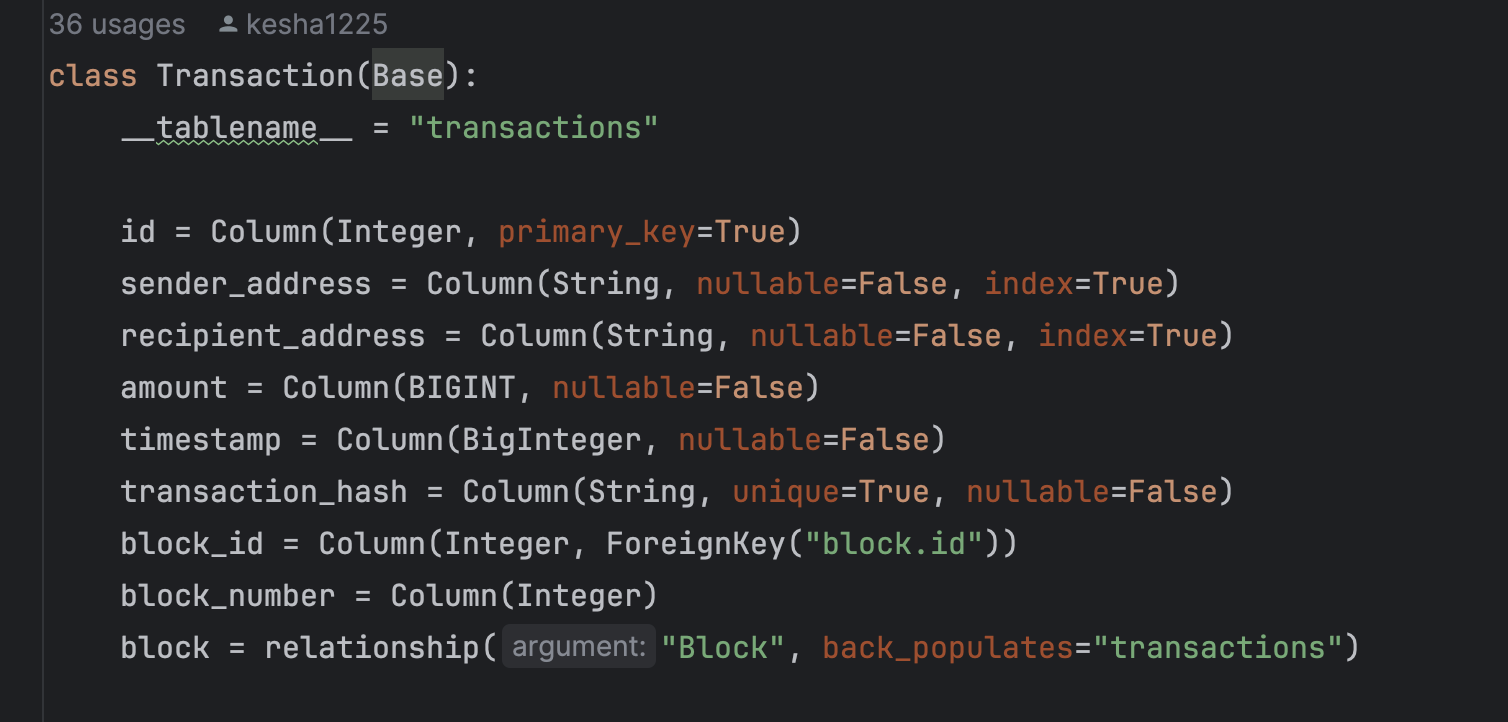


Рисунок 6 – структура транзакции в базе данных

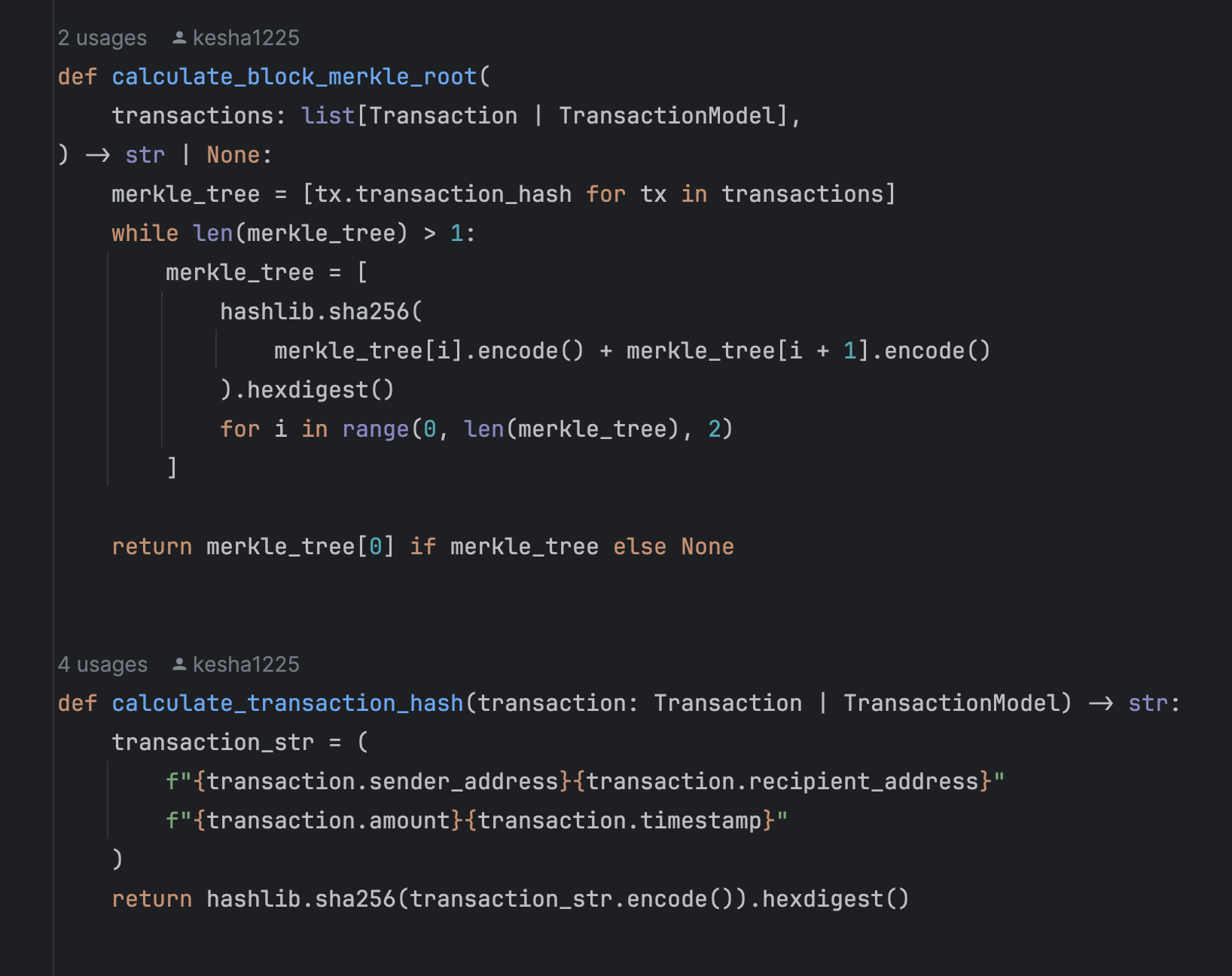


Рисунок 7 – Функции вычисления дерева меркла и хеша транзакции

Добавление случайного числа в транзакции блокчейна используется для предотвращения возникновения одинаковых хешей в случае, если две или более транзакции имеют одинаковые параметры. Это делает атаки на блокчейн сложнее, так как даже небольшие изменения в транзакции приведут к различным хешам блоков.

Однако, в этом случае я использовал более точное время до микросекунд вместо случайного числа, чтобы обеспечить уникальность хеша транзакции. Это значит, что каждая транзакция будет иметь уникальное время создания, которое до микросекунд точно отражает момент создания транзакции. Поскольку время до микросекунд является уникальным для каждой транзакции, нет необходимости в использовании случайного числа для предотвращения коллизий хешей.

Использование точного времени до микросекунд вместо случайного числа позволяет избежать хранения дополнительных данных в транзакциях и сделать их более легкими и эффективными. Такой подход также обеспечивает безопасность и уникальность каждой транзакции без необходимости генерации случайных чисел.

В базе данных для хранения блоков и транзакций использование индексов на полях sender\_address и recipient\_address является полезным с точки зрения оптимизации производительности и ускорения выполнения запросов. Индексы на эти поля позволяют значительно снизить время выполнения запросов, связанных с поиском или фильтрацией транзакций по отправителю или получателю.

Использование индексов на полях sender\_address и recipient\_address позволяет базе данных эффективно находить все транзакции, отправленные или полученные конкретным адресом. Это особенно полезно при выполнении запросов на поиск истории транзакций для определенного адреса или при анализе транзакций между определенными адресами.

Благодаря индексам на указанных полях база данных может быстро определить, какие записи соответствуют критериям запроса, что уменьшает время ответа и улучшает общую производительность системы. Такое оптимизированное хранение и доступ к данным важно для эффективной работы блокчейна, особенно при обработке большого объема транзакций.

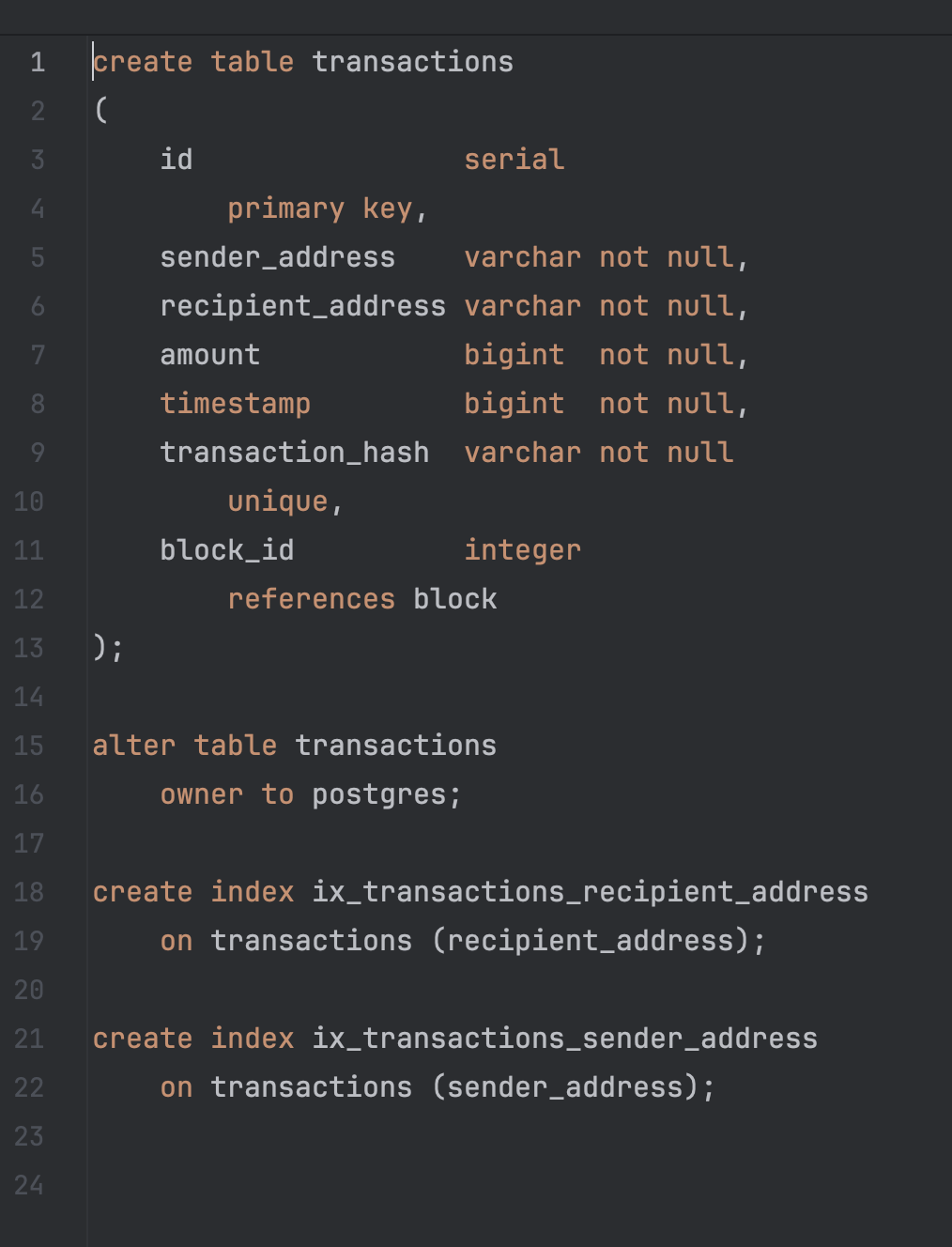


Рисунок 8 – Описание модели транзакции на DDL вместе с индексами на полях sender\_address и recipient\_address

Bech32 - это формат адресов, используемый в блокчейне, который представляет собой надежный и эффективный способ кодирования данных. Он основан на алгоритме Безье, который обеспечивает высокую степень надежности и устойчивости к ошибкам при передаче данных.

Одним из ключевых преимуществ Bech32 является его способность предотвращать ошибки при копировании и вводе адресов. Это достигается за счет использования символов, которые легко различимы человеческим глазом и имеют высокую степень различимости между собой. Это делает Bech32 идеальным выбором для блокчейна на PoA, где точность и надежность важны для успешной обработки транзакций.

Кроме того, Bech32 обеспечивает эффективное использование места и уменьшает размер передаваемых данных. Это особенно важно для блокчейна на PoA, где меньший размер транзакций приводит к более быстрой обработке и уменьшает нагрузку на сеть.

Таким образом, Bech32 является идеальным выбором для блокчейна на PoA благодаря своей высокой надежности, эффективному использованию места и способности предотвращать ошибки при передаче данных.

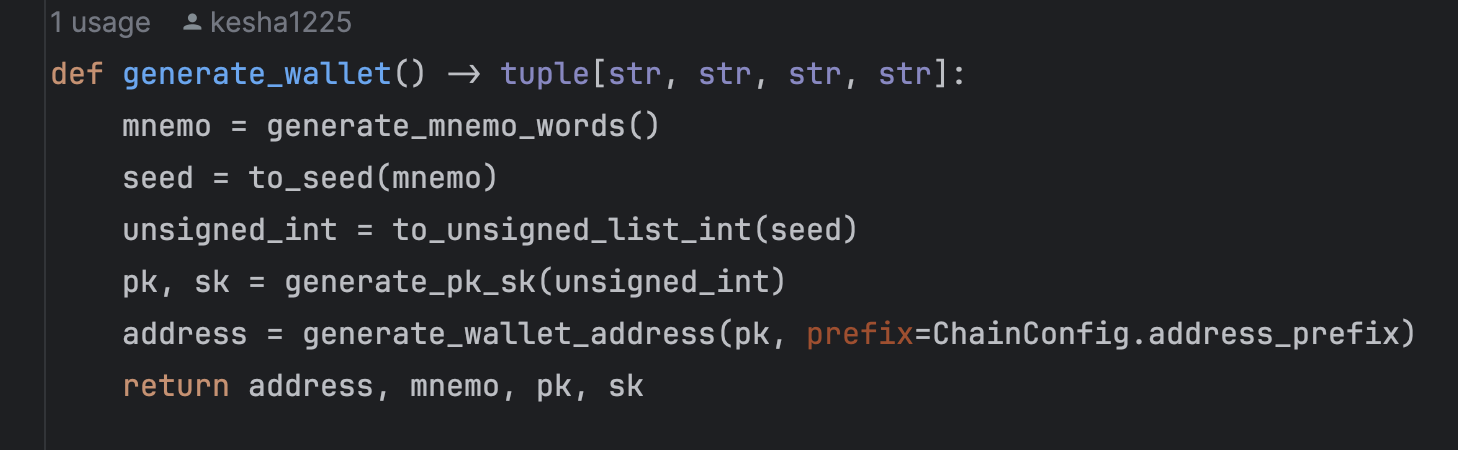


Рисунок 9 – функция генерации кошелька с префиксом poa с помощью bech32

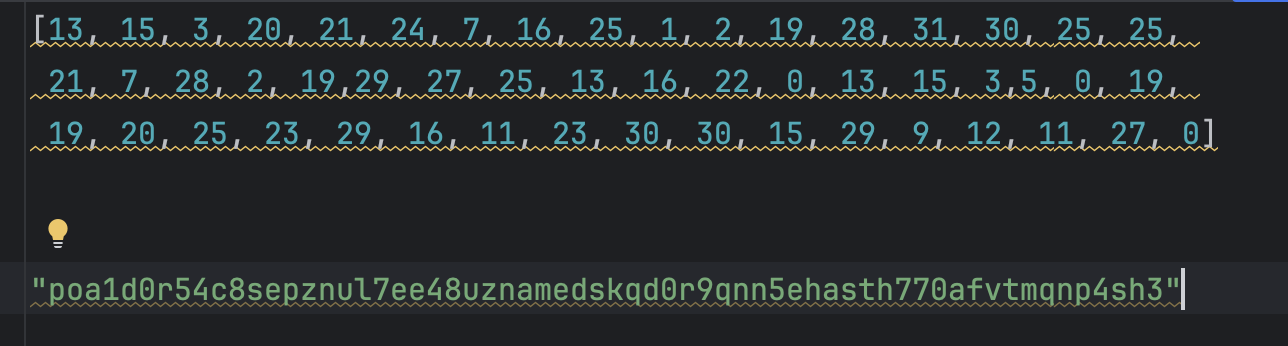


Рисунок 10 – Конвертированные байты из публичного ключа и адрес получившийся из них путем закодирования с помощью bech32

Описание модуля блокчейна, представленное выше, подчеркивает ключевые аспекты его разработки и функционирования. Рассмотрены основные компоненты блокчейна, включая структуру блока, транзакции, ноды и веб-интерфейс. Архитектура проекта основана на использовании СУБД PostgreSQL, клиентского веб-интерфейса и нод, что обеспечивает надежную и эффективную работу блокчейна.

Использование инновационных технологий, таких как Bech32 для адресации и точное время до микросекунд для обеспечения уникальности транзакций, подчеркивает стремление к оптимизации и улучшению производительности блокчейна. Такой подход отражает не только технический аспект проекта, но и его стремление к созданию надежной и эффективной системы для проведения трансграничных транзакций.

**3.2.3 Модуль ноды**

Модуль ноды – это сердце блокчейна. Он отвечает за создание, проверку и утверждение блоков, которые составляют основу цепи транзакций. Этот модуль также обеспечивает взаимодействие пользователей с блокчейном через веб-интерфейс.

Нода является ключевым элементом сети, так как именно здесь происходит процесс генерации новых блоков на основе подтвержденных транзакций. Кроме того, нода отвечает за поддержание согласованности и безопасности сети блокчейна.

Особенностью этого модуля является его распределенность. Ноды блокчейна могут быть развернуты в различных местах сети, что обеспечивает децентрализацию и надежность системы. Также нода способна взаимодействовать с другими узлами блокчейна для синхронизации данных и подтверждения блоков.

Таким образом, модуль ноды играет ключевую роль в функционировании блокчейна, обеспечивая его работу и взаимодействие с пользовательскими интерфейсами.